Fluxo de Controle - Estruturas de Decisão

APRESENTAÇÃO

Você já parou para pensar que, em nosso cotidiano, temos que tomar diversas decisões? Seja sobre a roupa que iremos utilizar, ou o que vamos almoçar ou jantar. Essas são situações em que devemos analisar as condições para que possamos escolher uma delas, praticamente escolhemos a que é mais viável com nossa situação atual. Esses mesmos passos ocorrem quando estamos elaborando um algoritmo, em que o mesmo deverá solucionar algum problema computacional e que, em algum momento, haverá mais de uma condição disponível.

Nesta Unidade de Aprendizagem, estudaremos sobre os conceitos e particularidades das estruturas de controle, as quais podem ser denominadas como simples, compostas e encadeadas. Veremos também a diferença entre suas sentenças e exemplos práticos por meio de pseudocódigo e da linguagem utilizada no MATLAB.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

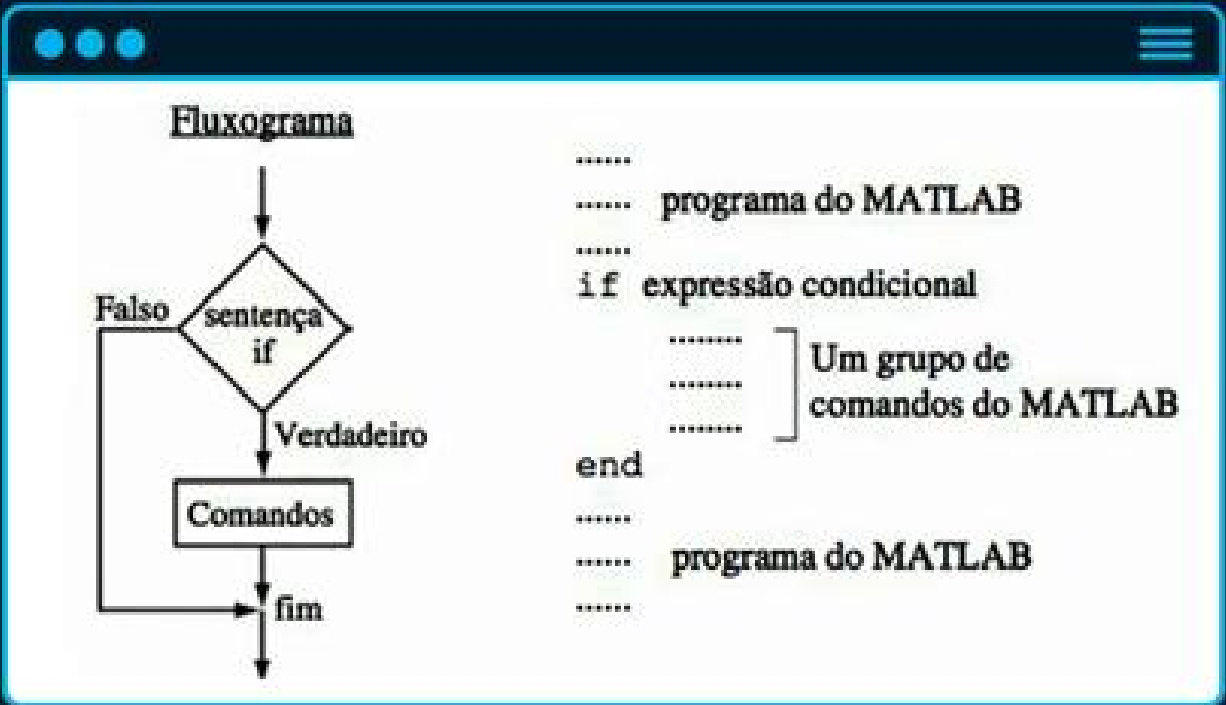
- Analisar os algoritmos com estruturas condicionais simples e/ou compostas em pseudocódigo e linguagem .m.
- Identificar problemas que necessitem de estruturas condicionais simples e/ou compostas.
- Desenvolver algoritmos em pseudocódigo e linguagem .m que necessitem estruturas condicionais simples e/ou compostas na resolução de problemas.

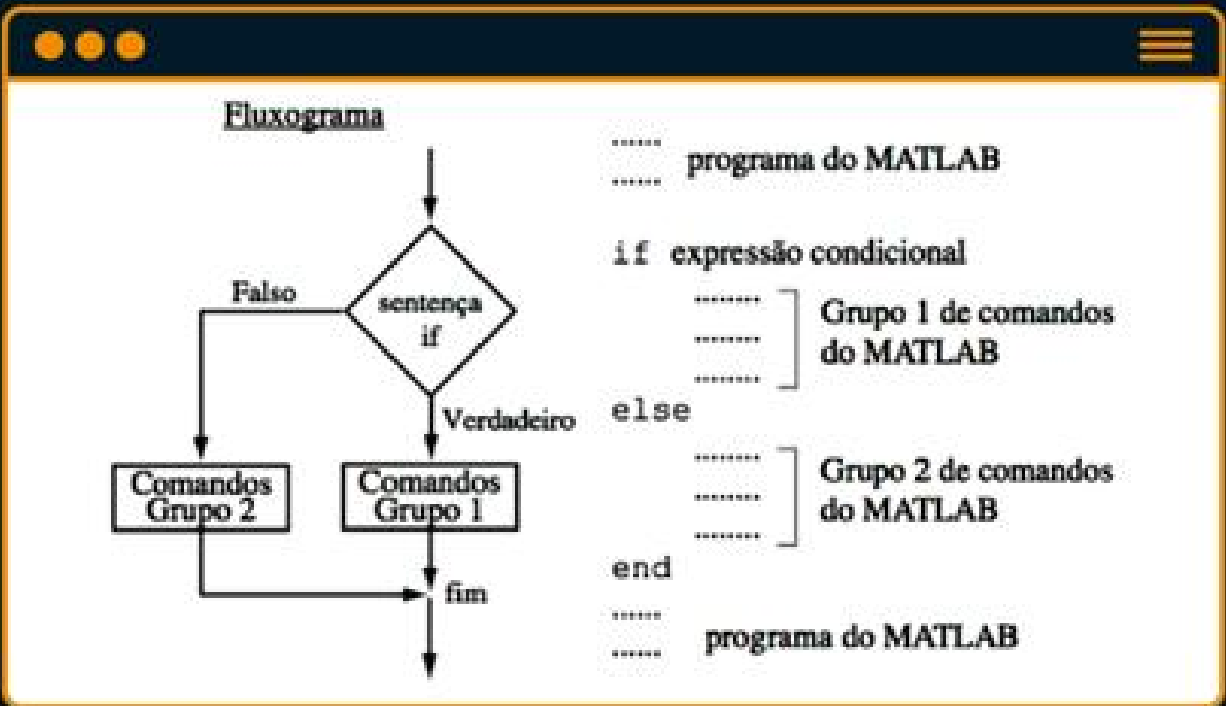
INFOGRÁFICO

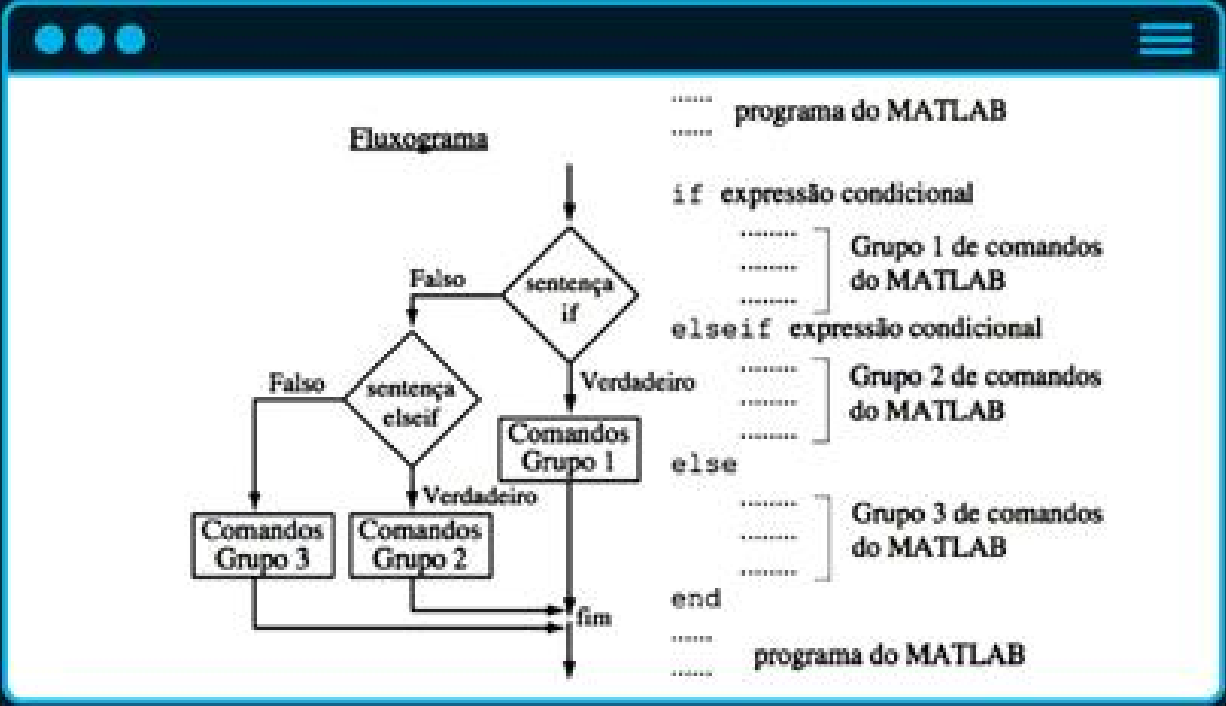
A solução de um problema, por meio do uso de um software, não se limita apenas à codificação. O uso do fluxograma é bastante eficiente para demonstrar a lógica aplicada na solução dos problemas, sem falar que eles auxiliam no momento da codificação do programa, independentemente da linguagem de programação, ou até mesmo com o uso do pseudocódigo.

O Fluxograma, simbolicamente, dita o fluxo ou a sequência segundo a qual os comandos serão executados. No Infográfico a seguir, você verá o fluxograma das sentenças if - end, if-else-end e if-elseif-else-end.

FLUXOGRAMAS DAS ESTRUTURAS DE CONTROLE

FLUXOGRAMA DA SENTENÇA if - end

Quando um programa é executado e encontra uma sentença if, se a expressão condicional de teste for verdadeira (1), o programa continua a executar os comandos listados abaixo da sentença if, até encontrar o comando end. Se a expressão de teste é falsa (0), o programa salta o grupo de comandos entre as sentenças if e end, continuando a execução dos comandos listados abaixo de end.

FLUXOGRAMA DA SENTENÇA if-else-end

A primeira linha contém uma sentença if com a respectiva expressão de teste. Se o resultado da expressão de teste é verdadeiro, o programa executa os comandos do grupo 1 (entre as sentenças if e else) e salta para o comando end. Se o resultado do teste é falso, o programa salta para a sentença else e executa os comandos do grupo 2 (entre as sentenças else e end).

FLUXOGRAMA DA SENTENÇA if-elseif-else-end.

Essa estrutura possui duas sentenças de teste (if e elseif) que tornam possível selecionar um dos três grupos de comandos para a execução. A primeira linha traz a sentença if com a expressão de teste. Se o resultado do teste é verdadeiro, o programa executa os comandos do grupo 1 (entre as sentenças if e elseif) e salta para end. Se o resultado do teste da sentença if é falso, o programa salta para a sentença elseif. Caso a expressão de teste de elseif seja verdadeira, o programa executa os comandos do grupo 2 (entre elseif e else) e salta para end. Se a expressão de teste de elseif é falsa, o programa salta para else e executa os comandos do grupo 3 (entre else e end).

CONTEÚDO DO LIVRO

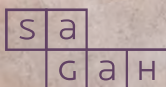
As estruturas de decisão trazem dois contextos, o primeiro é o de que a execução do programa só irá continuar caso a condição analisada seja verdadeira. Nesse caso, utilizamos a estrutura condicional simples, já que só teremos uma condição para analisar e, consequentemente, uma instrução, ou um único bloco de instruções para serem executados.

Para saber mais, leia o capítulo Fluxo de Controle - Estruturas de Decisão que faz parte do livro *Algoritmo e* programação e é base teórica desta Unidade de Aprendizagem.

Nele, você terá a oportunidade de compreender melhor o contexto que envolve as estruturas de controle, as quais, muitas vezes, são denominadas como condicionais. Iremos aprender que existem algumas sentenças utilizadas nele. Uma delas é a estrutura de controle simples (if -end), a estrutura composta (if-else-end) e a estrutura encadeada (if-elseif-else-end). Elas se diferenciam de acordo com a quantidade de condições que devemos testar para que o programa possa dar continuidade em sua execução.

Boa leitura.

ALGORITMO E PROGRAMAÇÃO

Izabelly Soares de Morais

Fluxo de controle: estruturas de decisão

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Analisar os algoritmos com estruturas condicionais simples e/ou compostas em pseudocódigo e linguagem MATLAB (.M).
- Identificar problemas que necessitem de estruturas condicionais simples e/ou compostas.
- Desenvolver algoritmos em pseudocódigo e linguagem MATLAB (.M) que necessitem estruturas condicionais simples e/ou compostos na resolução de problemas.

Introdução

Você já parou para pensar que em nosso cotidiano temos que tomar diversas decisões, seja sobre a roupa que iremos utilizar ou o que vamos almoçar ou jantar? Estas são situações em que devemos analisar as condições para que possamos escolher uma delas. Praticamente escolhemos a que é mais viável com nossa situação atual. Esse mesmo processo ocorre quando estamos elaborando um algoritmo que deverá solucionar algum problema computacional e que em algum momento haverá mais de uma condição disponível.

Neste capítulo, estudaremos sobre os conceitos e as particularidades das estruturas de controle, as quais podem ser denominadas como simples, compostas e encadeadas. Veremos também a diferença entra suas sentenças e exemplos práticos por meio de pseudocódigo e linguagem MATLAB.

Estruturas de decisão

Quando lidamos com situações comuns em nosso cotidiano, quase sempre nos deparamos com uma dúvida em relação a algo. Seja a respeito da nossa roupa, do nosso almoço ou até mesmo de qual caminho iremos seguir para ir até o

trabalho ou a faculdade. Você já imaginou como seria se só tivéssemos uma única opção para cada atividade? Nossa vida seria provavelmente um pouco monótona.

Dentro desse contexto, imagine se cada software fosse responsável por solucionar um único problema por vez. Por exemplo, caso fosse utilizar uma calculadora, teria que ter um programa para realizar a soma, outro para realizar a subtração, outro para a multiplicação, outro para a divisão e assim por diante. Com a quantidade de aplicativos que utilizamos hoje em dia, não haveria memória que comportasse essa quantidade absurda de softwares.

As estruturas condicionais, como o próprio nome já diz, nos dá a possibilidade de analisarmos condições, para que a partir dessa análise possamos dar instruções para que o programa execute. Perceba que estamos falando de software, sistemas, dessa forma, se falarmos utilizando nossa linguagem com o computador, ele claramente não irá compreender. Utilizaremos aqui o MATLAB, que, conforme Chapra (2013, p. 41), "é um software que fornece ao usuário um ambiente adequado à realização de diversos tipos de cálculos; além disso, contém ferramentas bastante úteis para implementação de métodos numéricos". No nosso contexto ele será utilizado para exemplificarmos os conceitos que forem vistos.

De acordo com Palm (2013, p. 160),

> [...] a programação estruturada é uma técnica para o projeto de programas em que uma hierarquia de módulos é utilizada, cada um tendo uma única entrada e um único ponto de saída, e em que o controle é passado de cima para baixo através da estrutura sem ramificações incondicionais para os níveis mais altos.

Ainda sob o ponto de vista do autor, a programação estruturada, se utilizada adequadamente, resulta em programas fáceis de serem escritos, entendidos e modificados. As vantagens da programação estruturada são as seguintes:

1. Os programas estruturados são mais fáceis de serem escritos porque o programador pode primeiramente estudar o problema como um todo e lidar com os detalhes posteriormente.
2. Os módulos (funções) escritos para uma aplicação podem ser utilizados em outras aplicações (isso é chamado de código reutilizável).
3. Os programas estruturados são mais fáceis de debugar porque cada módulo é projetado para realizar apenas uma tarefa, assim, ele pode ser testado separadamente dos outros módulos.

4. A programação estruturada é eficaz num ambiente de trabalho em equipe porque diversas pessoas podem trabalhar em um programa comum, cada pessoa desenvolvendo um ou mais módulos.
5. Os programas estruturados são mais fáceis de serem entendidos e modificados, especialmente se nomes significativos são escolhidos para os módulos e se a documentação identifica com clareza a tarefa de cada módulo.

O autor ainda complementa essas informações trazendo algumas outras questões, tais como a utilização de uma língua natural, como o português, para descrever algoritmos, resultando frequentemente em uma descrição verborrágica e sujeita a erros de interpretação. Para evitar lidar imediatamente com a sintaxe possivelmente complicada da linguagem de programação, podemos utilizar um pseudocódigo, no qual uma língua natural e expressões matemáticas são utilizadas para construir sentenças que se parecem com sentenças computacionais, mas sem a sintaxe detalhada. O pseudocódigo também pode utilizar alguma sintaxe simples do MATLAB para explicar a operação do programa. "O pseudocódigo é útil para esboçar um programa antes de o código ser escrito detalhadamente, o qual leva mais tempo para ser escrito, porque precisa estar em conformidade com as regras estritas do MATLAB" (PALM, 2013, p. 151).

Antes de falarmos sobre as estruturas, é relevante sabermos que diversos conceitos matemáticos são utilizados, já que o universo do algoritmo está envolto da solução de problemas, utilizando muitas vezes a lógica matemática. A Tabela 1, a seguir, traz os operadores relacionais utilizados.

Tabela 1. Operadores relacionais.

Operador relacional	Descrição
<	Menor que
>	Maior que
<=	Menor que ou igual a
>=	Maior que ou igual a
==	Igual a
~=	Diferente de

Além deles, fazemos uso também dos operadores lógicos, conforme se observa na Tabela 2.

Tabela 2. Operadores lógicos.

<table>
<tr><th>Operador lógico</th><th>Nome</th><th>Descrição</th></tr>
<tr><td>&
Exemplo: A&B</td><td>AND</td><td>Age em dois operandos (A, B). Se ambos forem verdadeiros, o resultado será verdadeiro (1). De outro modo, o resultado será falso (0).</td></tr>
<tr><td>|
Exemplo: A|B</td><td>OR</td><td>Age em dois operandos (A, B). Se um dos operandos for verdadeiro, ou se ambos forem verdadeiros, o resultado será verdadeiro (1). De outro modo (ou seja, ambos falsos), o resultado será falso (0).</td></tr>
<tr><td>~
Exemplo: ~A</td><td>NOT</td><td>Age em um operando (A). Resulta na negação do operando: verdadeiro (1), se o operando for falso, e falso (0), se o operando for verdadeiro.</td></tr>
</table>

Fonte: Gilat (2012).

As estruturas de decisão trazem dois contextos, o primeiro é o de que a execução do programa só irá continuar, caso a condição analisada seja verdadeira, neste caso utilizamos a estrutura condicional simples, já que só teremos uma condição para analisar, e consequentemente uma instrução, ou um único bloco de instruções para serem executados. O outro cenário se difere do primeiro, porque teremos uma instrução para ser executada tanto para caso a condição seja verdadeira quanto para caso ela seja falsa. A estrutura que possibilita que isso aconteça é a estrutura condicional composta. Além disso, temos a possibilidade de fazer a análise de mais de uma condição. Isso pode ocorrer por diversos motivos, um deles é: caso a primeira condição seja falsa e queiramos testar uma outra condição, estamos falando da estrutura que chamamos de encadeada, mas que no geral é uma estrutura composta encadeada. Veremos a seguir como elas podem ser utilizadas.

Fique atento

O MATLAB utiliza três janelas (*windows*) principais (CHAPRA, 2013):

- *Command window*: utilizada para inserir comandos e dados.
- *Graphics window*: utilizada para exibir gráficos.
- *Edit window*: utilizada para criar e editar arquivos-M (M-files ou programa/funções do MATLAB).

Aplicações práticas das estruturas de controle

Como citamos anteriormente, as estruturas condicionais se diferem conforme as possibilidades de execução dos trechos de código após a análise da condição. Existem estruturas condicionais simples, compostas e encadeadas, descritas a seguir.

Estrutura de controle simples

A estrutura condicional simples tem um bloco de instruções para serem executadas apenas caso a condição analisada seja verdadeira. Observe, na Figura 1 a seguir, como essa informação pode ser visualizada em forma de fluxograma.

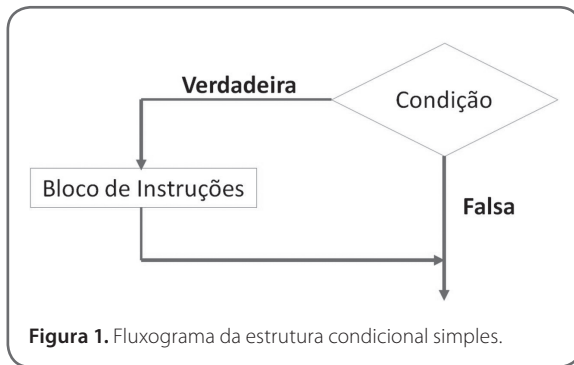
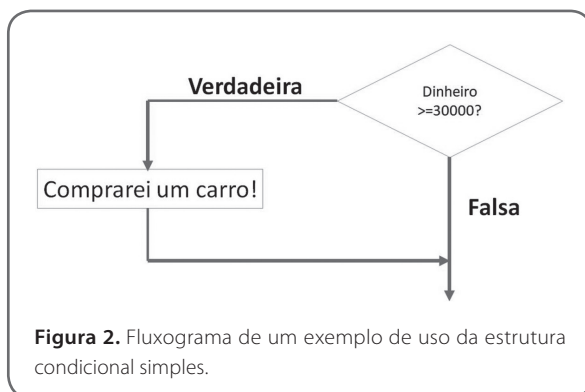



Figura 1. Fluxograma da estrutura condicional simples.

Devemos ter em mente que um fluxograma é utilizado para representar de maneira ilustrativa os possíveis caminhos que a execução de um software pode levar, ou seja, ele auxilia a compreensão da solução do problema e é útil para que seja elaborado antes do processo de codificação do programa, pois é muito mais simples alterar um fluxograma do que toda a codificação de um programa.

Imagine que você gostaria muito de adquirir um novo carro. Portanto, se você tiver dinheiro, você conseguirá comprar o carro; se você não tiver o dinheiro, você não comprará o carro e esse contexto deixa de existir. Observe que você só conseguirá comprar o carro se a condição de você ter dinheiro disponível for verdadeira. Como essa situação poderia ser exemplificada em forma de fluxograma? Veja na Figura 2, a seguir.

Figura 2. Fluxograma de um exemplo de uso da estrutura condicional simples.

Dessa forma, podemos notar que o bloco de instruções só é executado se a condição for atendida, ou seja, verdadeira. Aí você pode se perguntar: mas e se a condição analisada for falsa? Se ela for falsa, o programa termina a execução desse trecho de código e, dessa forma, as instruções após o **end** serão executadas sequencialmente ou finalizadas. Vamos representar esse contexto primeiramente utilizando uma linguagem natural, ou seja, nossa linguagem do cotidiano:

```
se (condição)
Instruções a serem executadas(caso a condição seja atendida)
fim
```

Quando utilizamos uma linguagem de programação, devemos seguir uma sintaxe. Nas estruturas de controle, utilizamos a palavra reservada (ou sentença) **if**, que quando traduzida para o português significa “se”, e **end**, que significa a finalização do bloco de instruções da estrutura de controle, ou seja, em complemento ao uso dessa palavra temos a seguinte sintaxe, que é utilizada na estrutura de controle simples:

```
if (condição)
Instruções a serem executadas(caso a condição seja atendida)
end
```

Mostramos o exemplo citado em um fluxograma. Agora, como ele poderia ser representado utilizando a sintaxe anterior? Vamos supor que o valor do carro é de R$ 30.000,00, ou seja, essa será a condição para que possamos comprar o carro. Veja como fica:

```
if dinheiro >= 30000
disp ('Comprarei um carro!')
end
```

Já que nosso exemplo diz que o carro custa R$ 30.000,00, utilizamos na condição o operador relacional >=, porque utilizando apenas o sinal de "maior que", não abrangeria o valor de R$ 30.000,00, e se tivéssemos só esse valor, sem nenhum valor a mais, a condição já estaria falsa. Por isso, colocamos o sinal de igualdade também, pois, caso tenhamos também esse valor, a condição será válida.

Estrutura de controle composta e encadeada

Falamos até agora apenas da possibilidade de termos um retorno, que, no caso, é a execução de instruções somente se a condição for verdadeira, mas e se houver a necessidade de ter um retorno caso ela seja falsa? Para isso, utilizamos a estrutura de controle composta. A Figura 3 traz a representação de um fluxograma dessa estrutura.

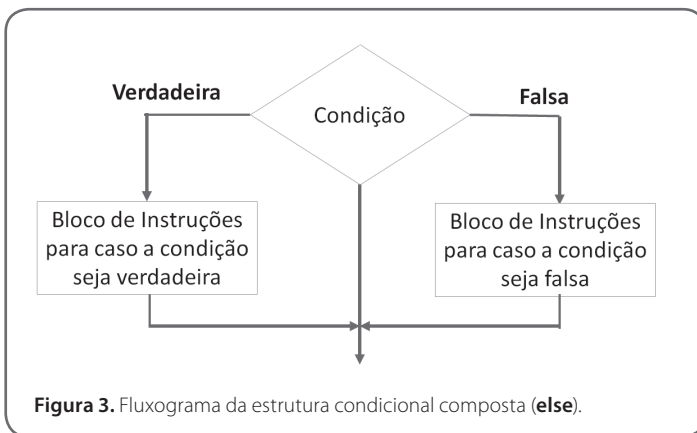
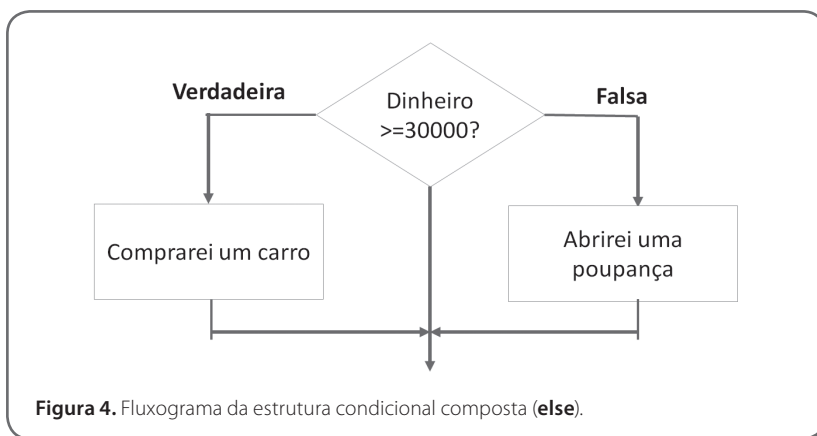
Figura 3. Fluxograma da estrutura condicional composta (**else**).

Vamos continuar com nosso exemplo de que você quer comprar o carro. Nesse novo contexto, temos uma nova condição. Portanto, se você tiver dinheiro, você conseguirá comprar o carro, e caso você não tenha dinheiro para comprar o carro, você irá iniciar uma poupança para juntar dinheiro, ou seja, se você não tiver o dinheiro, você não comprará o carro, mas o contexto não deixa de existir, já que você tem uma segunda opção, que é abrir a poupança. Como essa nova situação poderia ser exemplificada em forma de fluxograma? Veja na Figura 4.

Figura 4. Fluxograma da estrutura condicional composta (**else**).

Por meio do fluxograma, podemos observar que temos a possibilidade de obter mais de um retorno. Com isso, nossa sintaxe recebe novos comandos (sentenças). Além do **if**, temos também o **else**, que significa "senão", e o **elseif**, utilizado se a condição do **else** não for atendida e tivermos outra condição para ser analisada. A utilização fica da seguinte forma:

```
    se (condição)
    Instruções a serem executadas (caso a condição seja
    verdadeira)
    senão
    Instruções a serem executadas (caso a condição seja
    falsa)

fim
```

Os blocos de comando que fazem uso dessa sentença são denominados de encadeados. Já o pseudocódigo utilizando o **elseif** fica desta forma:

```
    se (condição 1)
    Instruções a serem executadas (caso a condição 1 seja
    verdadeira);
    senão
      se (condição 2)
      Instruções a serem executadas (caso a condição 2 seja
    verdadeira);
      senão
      Instruções a serem executadas (caso a condição 1 e a
    condição 2 sejam falsas);
      fimse
fim
```

E utilizando a sentença **elseif** fica desta forma:

```
if (condição 1)
disp (Instruções a serem executadas (caso a condição 1
seja verdadeira);
elseif (condição 2)
disp (Instruções a serem executadas (caso a condição 2
seja verdadeira);
end
```

Saiba mais

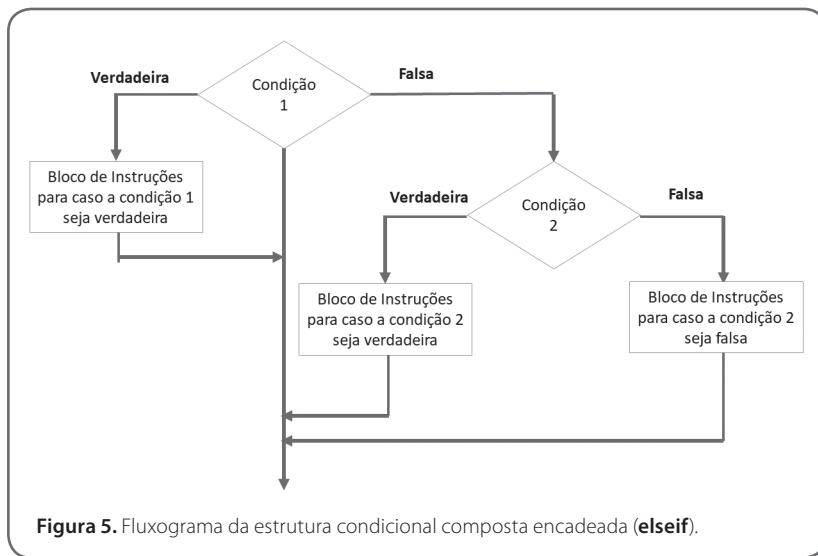
Muitas vezes, devemos tentar tornar nosso código mais conciso. Podemos utilizar alguns meios para realizar tal artimanha, como a mostrada no exemplo a seguir.

```
if condição lógica 1
      if condição lógica 2
Instruções para serem executadas caso as condições 1 e 2
sejam verdadeiras;
  end
end
```

Essas sentenças podem ser substituídas por um programa mais conciso, com o uso dos operadores lógicos, por exemplo.

```
if condição lógica 1 & condição lógica 2
Instruções para serem executadas caso as condições 1 e 2
sejam verdadeiras;
end
```

A sentença **elseif** já traz também outra representação do fluxograma, como observa-se na Figura 5.

Figura 5. Fluxograma da estrutura condicional composta encadeada (**elseif**).

Porém, você pode, ainda, pensar o seguinte: que você pode não ter dinheiro para comprar o carro, mas o dinheiro que você tem é suficiente para você adquirir uma moto, então, por que essa condição não surgiu no fluxograma, em vez de criar a poupança? Se tivéssemos feito isso, nosso algoritmo não estaria totalmente correto, porque, se a condição não for atendida, a instrução seria executada, mas e se você não tivesse nenhum dinheiro? Como iria comprar a moto? E se não tivesse dinheiro suficiente? Porém, há um modo de analisar essas condições. Vamos supor que uma moto custe mais que R$ 5.000,00 e menos que R$ 10.000,00. Dessa vez, faremos a demonstração utilizando a sintaxe da linguagem e aproveitaremos para utilizar os operadores lógicos:

```
If (dinheiro >=30000)
          disp ('Comprarei um carro!');
            elseif (dinheiro >=5000) & (dinheiro<=10000)
          disp ('Comprarei uma moto!');
          else
          disp('Abrirei uma poupança!');
          end
```

Nesse caso, observe que quando utilizamos as sentenças do MATLAB não precisamos utilizar um **end** para o **elseif**, como foi demonstrado quando utilizamos o pseudocódigo. Utilizamos o operador lógico **e(&)**, pois caso você tenha entre R$ 5.000,00 e R$ 10.000,00, você conseguirá adquirir a moto. A estrutura composta encadeada traz a possibilidade de mais de uma condição ser analisada. Não pense você que só podemos utilizar com duas condições. O limite das condições a serem analisadas depende muito do projeto, ou seja, do que o software irá precisar para solucionar o problema.

Saiba mais

Além das estruturas de controle, existem também as estruturas sequenciais, as quais são executadas seguindo uma certa ordem. As estruturas de repetição, muitas vezes chamadas também de iterativas, repetem a execução de um bloco de instruções até que certa condição seja atendida.

Exemplos de uso das estruturas de controle

Agora que já conhecemos a sintaxe e as sentenças utilizadas, podemos ver exemplos práticos de uso das estruturas de controle simples, composta e encadeada.

Exemplo prático com a estrutura de controle simples

Neste novo exemplo, vamos supor a seguinte situação: precisamos realizar a soma de dois números e devemos conferir se o resultado dessa soma será maior ou igual a 50. Se for maior, deverá retornar o valor da soma e, caso a condição não seja atendida, o programa deve encerrar sua execução. Primeiro, precisamos declarar variáveis para receber os valores que informaremos.

O fluxograma mostrará como podemos solucionar esse problema (Figura 6). É importante ficar atento sobre a existência de diversas formas de solucionar o mesmo problema, portanto, os nossos exemplos não são as únicas maneiras de solução.

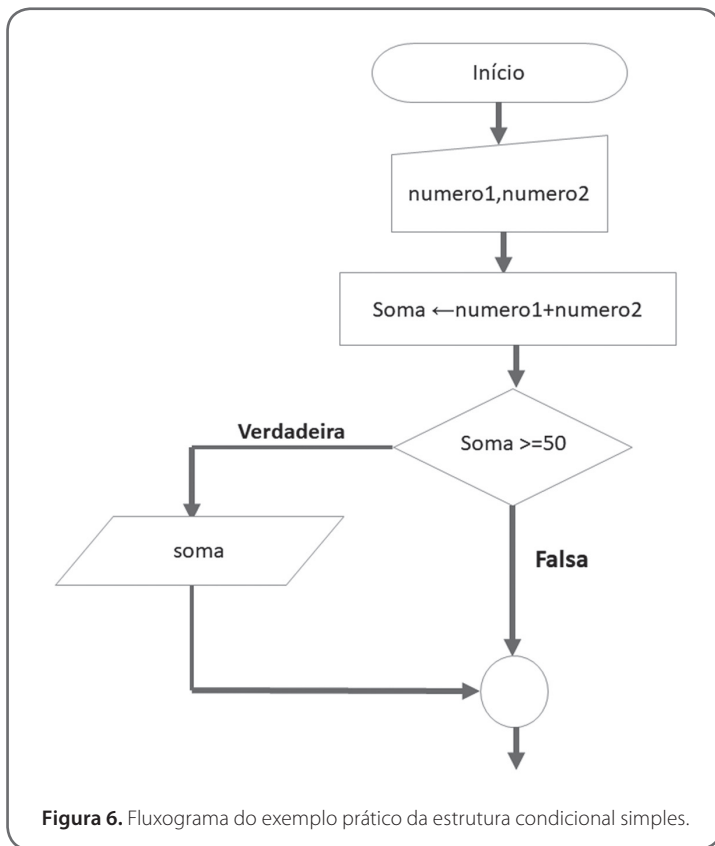



Figura 6. Fluxograma do exemplo prático da estrutura condicional simples.

Acompanhando essa mesma ideia e utilizando as sentenças utilizadas no MATLAB, temos:

```
numero1=input('Insira o primeiro número que deverá ser
somado');
numero2=input('Insira o segundo número que deverá ser
somado');
soma=numero1+numero2;
  if(soma>=50)
        fprintf('O valor da soma é: %d', soma);
  end
```

Não esqueça de que devemos sempre fazer uso de toda a sintaxe determinada pela linguagem de programação em que você estiver desenvolvendo o algoritmo. Aqui, estamos utilizando a linguagem aceita no MATLAB.

Exemplo prático com a estrutura de controle composta

Como estamos avançando nas estruturas, vamos continuar com o mesmo exemplo, porém agora iremos utilizar a estrutura de controle composta. Teremos de fazer a soma de dois valores, o que muda agora é que, ao invés de termos apenas a condição de que o valor obtido como resultado dessa soma seja maior ou igual a 50, vamos definir que caso essa condição seja aceita, o valor da soma deverá ser somado com 2; caso o valor somado não seja maior ou igual a 50, o valor da soma deve ser subtraído por 10.

Assim como fizemos anteriormente, vamos mostrar primeiro o fluxograma deste algoritmo (Figura 7).

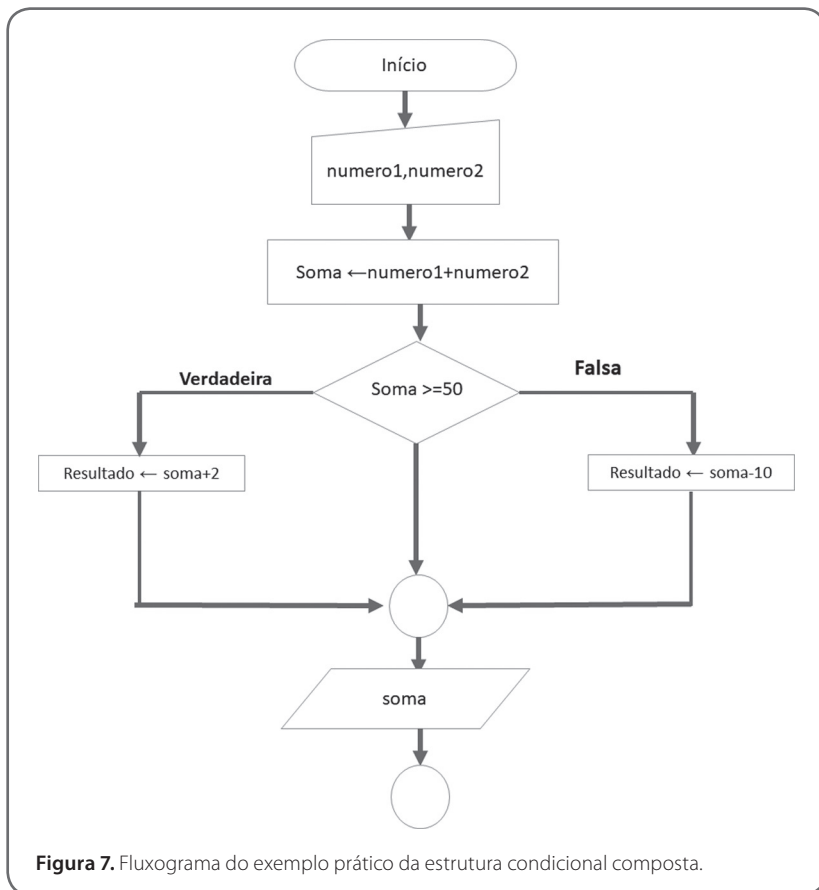



Figura 7. Fluxograma do exemplo prático da estrutura condicional composta.

Acompanhando essa mesma ideia e utilizando as sentenças utilizadas no MATLAB, temos:

```
numero1=input('Insira o primeiro número que deverá ser
somado');
numero2=input('Insira o segundo número que deverá ser
somado');
soma=numero1+numero2;
  if(soma>=50)
  resultado = soma+2;
  else
  resultado = soma-10;

  end
fprintf('O valor da soma é: ', resultado);
```

Fazendo um comparativo das sentenças, você pode perceber que a impressão do resultado só será realizada após a análise da condição. Isso ocorre porque agiliza a exibição do resultado, já que não estaria errado inserir essa impressão dentro de cada instrução, mas a melhor prática é inserir depois.

Exemplo prático com a estrutura de controle composta encadeada

Agora, ainda teremos que fazer a soma de dois valores, o que muda é que, ao invés de termos apenas a condição de que o valor obtido como resultado dessa soma seja maior ou igual a 50, caso essa condição seja falsa, devemos conferir também se o resultado da soma é menor ou igual a 100. Caso seja, devemos subtrair 8 da soma, e caso não seja, devemos somar 8 com a soma. Porém, caso a primeira condição seja aceita, o valor da soma deverá ser somado com 2.

A seguir, podemos acompanhar o fluxograma deste algoritmo (Figura 8).

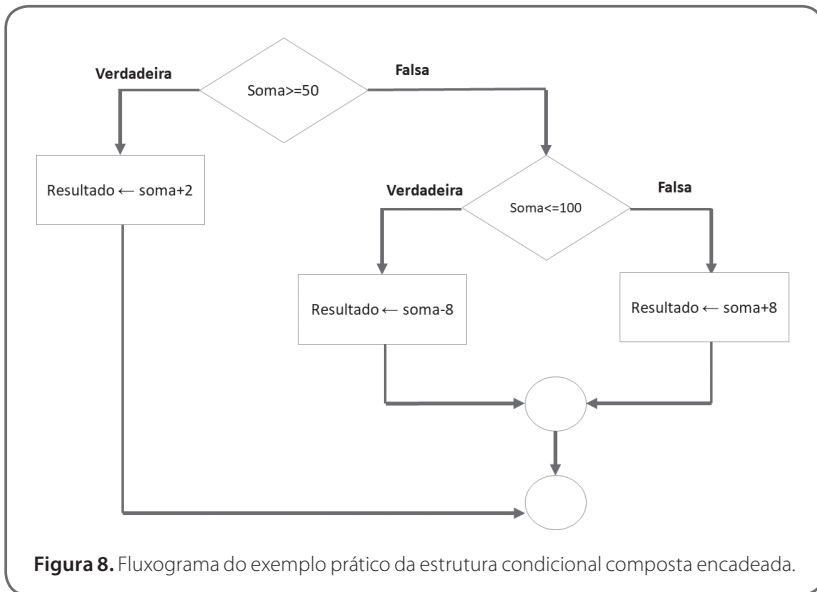



Figura 8. Fluxograma do exemplo prático da estrutura condicional composta encadeada.

Acompanhando essa mesma ideia e utilizando as sentenças utilizadas no MATLAB, temos:

```
numero1=input('Insira o primeiro número que deverá ser somado');
numero2=input('Insira o segundo número que deverá ser somado');
soma=numero1+numero2;
   if(soma>=50)
   resultado = soma+2;
   else if  (soma<=100)
   resultado = soma-8;
   else
   resultado = soma+8;
   end
fprintf('O valor do resultado é: ', resultado);
```

Uma observação importante é a de que o modo de desenvolver um algoritmo é particular, ou seja, provavelmente cada pessoa desenvolve sua identidade com relação ao modo como elabora seu algoritmo, desde que siga as regras estabelecidas pela linguagem de programação. Dessa forma, existem várias maneiras de solucionar o mesmo problema, aos poucos você vai descobrindo qual é o seu.

Referências

CHAPRA, S. C. *Métodos numéricos e aplicados com MATLAB® para engenheiros e cientistas*. 3. ed. Porto Alegre: AMGH, 2013.

GILAT, A. *MATLAB com aplicações em engenharia*. 4. ed. Porto Alegre: Bookman, 2012.

PALM, W. J. *Introdução ao MATLAB para engenheiros*. 3. ed. Porto Alegre: AMGH, 2013.

Leituras recomendadas

ASCENCIO, A. F. G.; CAMPUS, E. A. V. *Fundamentos da Programação de Computadores*. 3. ed. São Paulo: Pearson, 2012.

FORBELLONE, A. L. V. *Lógica de programação:* a construção de algoritmos e estrutura de dados. 3. ed. São Paulo: Pearson, 2005.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. *Algoritmos:* lógica para desenvolvimento de programação de computadores. 28. ed. São Paulo: Érica, 2016.

SEBESTA, R. W. *Conceitos de linguagens de programação*. 9. ed. Porto Alegre: Bookman, 2011

DICA DO PROFESSOR

O desenvolvimento de algoritmos depende muito de qual problema este sistema que será desenvolvido irá solucionar. O que devemos ter em mente é que temos diversas possibilidades de como isso pode ser feito.

Acompanhe na Dica do Professor a sintaxe das estruturas de controle simples, composta e encadeada, para você compreender melhor como cada uma funciona.

Conteúdo interativo disponível na plataforma de ensino!

EXERCÍCIOS

1) Diversos problemas exigem que as condições sejam analisadas. Para isso, existem alguns tipos de estrutura de controle, assinale a alternativa que traz os conceitos corretos sobre a estrutura de controle simples.

A) As sentenças if-end representam a estrutura de condição simples.

B) As sentenças if-enf-if-end trazem a sintaxe da estrutura controle simples.

C) As sentenças if-then-elseif-end trazem a representação da estrutura controle simples.

D) As sentenças if-else-end trazem a estrutura de controle simples.

E) As sentenças else-if-else-end trazem a estrutura da estrutura de controle simples.

2) A estrutura de controle composta, executa uma instrução conforme a análise da condição, a qual pode ser verdadeira ou falsa. Assinale a alternativa correta de acordo com a sintaxe na estrutura composta.

A) Ela é composta pelas sentenças if-else if-end.

B) Ela é composta pelas sentenças if-elseif – elseif-end.

C) Ela é composta pelas sentenças elseif – else-end.

D) Ela é composta pelas sentenças elseif – if-end-else if.

E) Ela é composta pelas sentenças if else if – end – end.

3) **Um algoritmo pode ser representado por diversas formas, marque a alternativa que traz o tipo de representação que pode ser feito por meio de figuras geométricas, as quais cada uma possui um significado coerente com a execução de um algoritmo.**

A) Pseudocódigo.

B) Fluxograma.

C) Java.

D) MATLAB.

E) Linguagens formais de programação.

4) **Em um determinado contexto, temos mais de uma condição a ser analisada, em que a segunda condição só será analisada caso a primeira seja falsa. Assinale a alternativa que traz a estrutura de controle e suas respectivas sentenças em pseudo-código que devem ser utilizadas para solução desse problema.**

A) Estrutura de controle simples, com as sentenças “se-senão-se-fim_se”.

B) Estrutura de controle simples e composta, com as sentenças “se-senão”.

C) Estrutura de controle composta, com as sentenças “se-senão-senão-end”.

D) Estrutura de controle encadeada, com as sentenças “senão-end”.

E) Estrutura de controle encadeada, com as sentenças “se-senão-se-senão-fim_se-fim_se”.

5) Para testar a lógica de um programa, podemos utilizar, além das estruturas de controle, os operadores lógicos. Em uma determinada situação, a instrução, para caso a condição seja verdadeira, só poderá ser executada caso o número seja maior ou igual que 30 e se o número for menor ou igual a 80. Assinale a alternativa que traz as sentenças que solucionam esse problema.

A) se(numero>30 e numero=30) e (numero<80 e numero=80).

B) se(numero>30 e numero <80).

C) se(numero >=30) e (numero<=80).

D) se(numero>30 e numero=30) ou (numero<80 e numero=80).

E) se(numero>30 ou numero <80).

NA PRÁTICA

Na prática, podemos aplicar os conceitos que norteiam os algoritmos e as linguagens de programação em todas as nossas atividades cotidianas. Na imagem a seguir, você poderá ver uma situação prática em que é demonstrado como, em uma situação cotidiana, pode ser exemplificada por meio do uso da estrutura de controle.

Na escolinha de futebol ABC, **os times dos alunos são formados conforme a faixa etária**. Isso ocorre para que o nível do jogo fique uniforme. A tabela de classificação está abaixo:

IDADE	CATEGORIA
DE 05 A 10 ANOS	INFANTIL
DE 11 A 15 ANOS	JUVENIL
DE 16 A 20 ANOS	JÚNIOR
DE 21 A 25 ANOS	PROFESSIONAL
DE 26 A 70 ANOS	SÊNIOR

PARA ISSO, FOI DESENVOLVIDA UMA RESOLUÇÃO EM PSEUDOCÓDIGO:

```
algoritmo

var
idade:inteiro

inicio

escreva("Digite a Idade: ")
leia (idade)
se (idade >=5) e (idade<=10) entao escreva("INFANTIL") senao se
(idade>=11) e (idade<=15) entao escreva("JUVENIL") senao se
(idade >=16) e (idade<=20) entao escreva("JÚNIOR") senao se
(idade>=21) e (idade<=25) entao escreva("PROFISSIONAL") senao
se (idade>=26) e (idade<=70) entao
escreva("SENIOR")
fimse
fimse
fimse
Fimse
 fimse
```

E UMA EM RESOLUÇÃO COM A LINGUAGEM M.

```
1 -    clear;
2 -    clc;
3
4 -    nome = input('Nome do Atleta: ', 's');
5 -    idade = input('Idade do Atleta: ');
6
7 -    if ((idade>=5)&&(idade<=10))
8 -        classificacao = 'infantil';
9 -        else if ((idade>=11)&&(idade<=15))
10 -       classificacao = 'juvenil';
11 -       else if ((idade>=16)&&(idade<=20))
12 -       classificacao = 'junior';
13 -       else if ((idade>=21)&&(idade<=25))
14 -       classificacao = 'profissional';
15 -       else if ((idade>=26)&&(idade<=70))
16 -       classificacao = 'senior';
17 -       else
18 -       classificacao = 'Não definida';
19 -           end
20 -           end
21 -           end
22 -           end
23 -   end
24 -   fprintf('\nCategoria do atleta %s: [%s].\n', nome, classificacao);
```

RETORNO DA EXECUÇÃO DO PROGRAMA.

```
Command Window
  Nome do Atleta: João
  Idade do Atleta: 35

  Categoria do atleta João: [senior].
fx >>
```

SAIBA MAIS

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Lógica de Programação Aula 20: Estrutura de Decisão Simples

No *link* a seguir, você terá acesso a um vídeo que traz a explanação de todos os conceitos que envolvem as estruturas de controle simples.

Conteúdo interativo disponível na plataforma de ensino!

Lógica de Programação Aula 20: Estrutura de Decisão Simples

No *link* a seguir, você terá acesso a um vídeo que traz uma aplicação prática das estruturas de controle simples.

Conteúdo interativo disponível na plataforma de ensino!

Lógica de Programação Aula 25: Exercício aluno aprovado ou reprovado

No *link* a seguir, você terá acesso a um vídeo que, dando continuação às estruturas de controle, este vídeo traz as particularidades da estrutura de controle composta.

Conteúdo interativo disponível na plataforma de ensino!

Lógica de Programação Aula 27: Estruturas de Decisão Aninhadas

No *link* a seguir, você terá acesso a um vídeo que traz os conceitos da estrutura de controle encadeada, as quais também podem ser denominadas como “aninhadas”.

Conteúdo interativo disponível na plataforma de ensino!